ADVISORY AND TRAINING SYSTEM FOR INTERNET-RELATED PRIVATE INTERNATIONAL LAW

D3.5 – InterLex integrated rule-base (including legislation, and case law (Final version))

Grant agreement number	800839
Project name/acronym	InterLex
Project title	Advisory and Training System for Internet-related Private International Law
Website	www.interlexproject.eu
Contractual delivery date	29/02/2020
Actual delivery date	25/06/2020
Contributing WP	3
Dissemination level	Public
Deliverable leader	UNIBO
Contributors	UNITO,APIS

This project has received funding from the European Union's Justice Programme (2014-2020) under Grant Agreement Number 800839

Document History

Version	Date	Author	Partner	Description
0.1	01/02/2020	Roberta Calegari Giuseppe Contissa Giovanni Sartor	UNIBO	Rulebase of Italian PIL - Initial version
0.2	10/03/2020	Roberta Calegari Giuseppe Contissa Giovanni Sartor	UNIBO	Rulebase of Italian PIL – final version
0.3	02/04/2020	Roberta Calegari Giuseppe Contissa Giovanni Sartor	UNIBO	Rulebase of Czech PIL – initial version
0.4	05/05/2020	Giuseppe Contissa	UNIBO	First complete draft of the Deliverable
0.5	20/05/2020	Roberta Calegari Giuseppe Contissa Giovanni Sartor	UNIBO	Revised internal version
0.6	30/05/2020	Roberta Calegari Giuseppe Contissa Giovanni Sartor	UNIBO	Second revised internal versions
1.0	13/06/2020	Roberta Calegari Giuseppe Contissa Giovanni Sartor	UNIBO	Final version

Contributors

Partner	Name	Role	Description
APIS	Hristo Konstantinov	Reviewer	All sections
UNITO	Luigi Di Caro	Reviewer	All sections
UNITO	Llio Humphreys	Reviewer	All sections

Table of Contents

List of Figures

N/A

List of Tables

N/A

List of Acronyms

PIL	Private International Law
UNIBO	UNIversity of BOlogna
WP	Work Package

Executive Summary

This document constitutes Deliverable 3.5 of the InterLex project. Its primary objective is to provide a report on the development of an integrated rule-base (including legislation, and case law) in the form of Prolog rules, consisting of a set of computable representations of the main EU regulations, national legislations and case law in the area of private international law. The integrated rule-base will constitute the core content of the Decision Support Module (DSM) of the InterLex Platform.

Section 1 introduces the application domain of PIL, including EU regulations and national laws. It describes the motivations for the selection of parts of legislation to be modelled.

Section 2 describes the structure and content of the rule-base, and the methodology adopted for its development. It is divided into 3 main subsections.

Subsection 2.1 provides an overview of the Prolog representation, covering the top-level structure of the rule-base, and including high-level representations of each of the three main EU source regulations and the currently available national representations (Italy and Czech Republic), as well as methods for enabling linking between the representations of EU regulations and national laws.

Subsection 2.2 explains the methodological approach followed in the development of the rule-base, including in particular the development of predicates, the management of exceptions, and the development of a meta-interpreter to provide explanations (the rationale for the assessment of the goal).

Subsection 2.3 presents a set of use case scenarios and a demonstration of the functioning of the Prolog system. In particular, it is meant to demonstrate the functioning of the system with hypothetical scenarios of EU law and national law.

Section 3 presents the conclusions, summarizing the most relevant parts of all the work done and the way forward for the final development and integration of the representations into the Interlex platform.

The deliverable is completed with an appendix containing (as an embedded document) the complete rule-base in Prolog, including the Prolog meta-interpreter for explanations.

1. Introduction: the EU and national legislation corpora for PIL

This deliverable presents the release of the integrated Interlex rule-base, in the form of Prolog rules. The rule-base consists of a computable representation of the main EU regulations in the area of private international law, and a set of representations for national legislation and case law.

With regard to the EU legal corpus, the representation covers 3 main legislative EU instruments: Regulation (EU) No. 1215/2012 on jurisdiction and the recognition and enforcement of judgments in civil and commercial matters (recast) (the Brussels Regulation); Regulation (EC) No. 593/2008 on the law applicable to contractual obligations (Rome I); and Regulation (EC) No. 864/2007 on the law applicable to non-contractual obligations (Rome II). The reference documents adopted as sources for the representation were the digital copies in PDF format and in the English language available on the EUR-Lex website.

With regard to the national level, the current version of the rule-base covers the main instrument of the Italian PIL law (Law of 31 May 1995, No. 218) and of the Czech PIL law (Act No. 91/2012 Coll., Private International Law Act).

The reference documents adopted as sources for the representation were the digital copies in PDF format and in the English language made directly available (or whose links were made available) on the InterLex Platform.

As a general rule, the articles relevant for the assessment of jurisdiction and applicable law have been represented. The missing articles contain either clarifications or examples (e.g. non-closed definitions). Representation of these rules will be evaluated for future releases of the rule-base.

Then, each norm from the core parts of the legal source texts was represented in a corresponding rule (or set of rules) by specialist knowledge engineers supported by legal experts. The legal analysis often involved the interpretation of complex legal rules, and the selection of one interpretation among several possible.

No deviations were made, as long as possible, from the original structure of the text, even when the resultant logic representation was redundant or convoluted), in order to preserve isomorphism (see InterLex Deliverable 3.1, section 1).

The legal analysis of the source laws was supported by use of legal commentaries, namely [1] , [2], and [3].

In the following subsections, we shall describe the legal corpora selected for the EU and national level representations.

1.1. EU LEGISLATION FOR PIL

The development of the representation started with legal analysis of the EU regulations, aimed at identifying the core part of laws to be represented in relation to the goals of the InterLex project. The following parts were selected:

- the Brussels Regulation: the core part of the regulation selected for the representation includes all the sections that are relevant for the identification of jurisdiction, namely Chapter I (Scope and Definitions); Chapter II (Jurisdiction), Sections 1 – 7; and Chapter V (General Provisions), articles 62, 63 and 65.

- the Rome I Regulation: the whole content of the regulation was selected for the representation, with the exception of article 18 and articles 21-25.

- the Rome II Regulation: the whole content of the regulation was selected for the representation, with the exception of articles 15-18, 21-22, and 26-31.

1.2. NATIONAL LEGISLATION FOR PIL

For the initial release of the integrated rule-base, Italian and Czech national legislation were selected because the semi-official translation of the national source laws were easily accessible. Further representations of national legislation will be progressively added.

With regard to the Italian legal system, the main legislative instrument in the field of private international law is the Law of 31 May 1995, No. 218 entitled "*Riforma del sistema italiano di diritto internazionale privato*" ("the Italian PIL Act"), which sets out the Italian system for private international law. It includes provisions regulating the general part of private international law (preliminary issues, *lis pendens*, *iura novit curia*, renvoi, application of foreign law etc.), general and special rules on jurisdiction, conflict-of-laws rules, and rules on recognition and enforcement[1].

For the purposes of the representation, the core part selected for the representation includes the following articles: Title II (Italian Jurisdiction), articles 3, 4, 5, 9, 12; Title III (Applicable Law), Chapter I to III, articles 19, 20, 22, 23, 24, 25; Title III Chapter IV (Family Relationships), articles 26, 27, 28, 29; Title III Chapter VIII (Rights in property), articles 51, 52, 53, 54, 55; Title III Chapter IX, article 56; Title III Chapter X, articles 58, 59, 60, 61, 62, 63.

With regard to the Czech legal system, the main legislative instrument is Act No. 91/2012 Coll., Private International Law Act (the 2012 PILA). It includes general provisions regulating its relationship to international treaties and EU regulations, mandatory rules, evasion of law or public order, and other general provisions of private international law. The most comprehensive part is Part Four (§§ 29-101), which contains provisions for

[1] For further legal analysis of the Italian system of Private International Law, see Interlex Deliverable D2.2 (Analysis of the data collected on a national basis), pp. 125-142.

individual types of private law relationships, such as contracts or delicts, and includes conflict-of-laws rules, rules on jurisdiction, and rules on recognition and enforcement of foreign judgments with regard to individual matters (legal capacity, family matters, rights in rem, succession, contractual and non-contractual obligations etc.)[2].

For the purposes of the representation, the core part selected for the representation includes articles from Part Four, Chapter V (§§ 47-52) concerning family law, and articles from Chapter VII (§§ 78-62) concerning rights in rem.

2. The Prolog representation

2.1. HIGH-LEVEL STRUCTURE OF THE RULE-BASE

The language adopted for the representation is SWI-Prolog, in accordance with the selection made at the beginning of the project and documented in Deliverable 3.1. However, the similarity of the SWI-Prolog implementation to the ISO-Prolog standard also means that the Prolog code is not entirely specific to the chosen Prolog implementation, but can be easily re-used with other versions of the language. The approach adopted for the modelling is described in Deliverable 3.1, section 3.4.

The InterLex Rule-Base base is logically organized according to the specific chapters and sections of the source legal documents selected for the representation. This is mirrored in the code base by having the different sections split into separate files.

The following are the main goals and the entry levels for each of the represented sources:

- the Brussels Regulation:

```
hasJurisdiction(Country, Court, ClaimId, Law)
```

- the Rome Regulation:

```
applicableLaw(Article, Country, ContractId, Law)
```

- the Rome II Regulation:

```
applicableLawNonContract(Article, Country, ObligationId, Law)
```

The top-level goal is immediately followed by the main exceptions and conditions that must be verified before importing the rules contained in other chapters and sections. As an example, this is a fragment from the Brussels Regulation representation:

```
hasJurisdiction(Country, Court, ClaimId, brusselsRegulation):-
```

[2] For further legal analysis of the Czech system of private international law, see InterLex Deliverable D2.2 (Analysis of the data collected on a national basis), pp. 69-77.

```
    brusselsRegulationApplies(ClaimId, brusselsRegulation),
    \+exception(hasGeneralJurisdiction, _, ClaimId),
    hasGeneralJurisdiction(Country, Court, ClaimId, brusselsRegulation).
```

In this example, the top-level goal (the assessment of the jurisdiction according to the Brussels Regulation) can be verified if: the premises (contained in Chapter 1 of the Regulation) apply, there are no exceptions to the general jurisdiction rules, and at the same time the general jurisdiction rules apply. The same approach has been adopted in the representations of the Rome I and Rome II Regulations.

The same top-level goals are used in the structure of the national legislation. The relevant sections from the Italian law (No. 218 – 1995) have been modelled, and are automatically linked to the EU law. In the EU rule-base, a couple of articles invoke the national law of a certain state.

```
hasJurisdiction(Article, Country, Court, ClaimId, nationalLaw(Country))
```

This predicate will find its solution in the relevant national rule-base, selected with the Country variable. All the national rule-bases are defined starting with the following main predicates:

```
hasJurisdiction(Article, Country, Court, ClaimId, national_law(italy)):
    hasJurisdictionPilItaly(Article, Country, Court, ClaimId, law(italy)).
applicableLaw(Article, Country, ClaimId, national_law(italy)):
    applicableLawPilItaly(Article, Country, ClaimId, law(italy)).
```

Having a generic predicate enables the abstraction of different legal texts within a single entry point.

The following is a simple example that shows how the representation mirrors the previous code, maintaining consistency in the expression of exceptions and variables.

```
hasJurisdictionPilItaly(article3, italy, Court, ClaimId,
nationalLaw(italy)):-
    \+exception(hasJurisdictionPilItaly(_, italy, _, ClaimId,
nationalLaw(italy), _),
    personRole(PersonId, ClaimId, defendant),
    personDomicile(PersonId, italy, Court).
```

Another simple example, this time related to the Rome Regulation (in the Italian legislative corpus both are contained in the same law), can be found in article 25, that states:

"*Si applica, tuttavia, la legge italiana se la sede dell'amministrazione è situata in Italia*"

(However, Italian law applies if the establishment is located in Italy)

```
applicableLawPilItaly(art25, italy, ClaimId, nationallaw(italy)):-
    personNature(PersonId, legal),
    personEstablishment(PersonId, italy, _).
```

These are simple examples, with only a couple of conditions, but they are representative of the law in question.

Using this structure, it is possible to invoke all different (and pertinent) national and international legislation, by replacing nationalLaw(Country) with a variable:

```
hasJurisdiction(Country, Court, claim, Law)
```

This is however impractical, since the interpreter does not know which legislation are applicable to the case under consideration. It is thus recommended to inpute manually, or by other logic statements, the relevant sources.

2.2. THE APPROACH TO THE MODELLING

2.2.1. Conversion process

The process of converting the initial legal text into rules necessarily involved several different stages, as required by commonly adopted Prolog-based rule-base creation processes.

This work was carried out in a sequence of iterative stages. The first stage consisted in the development of what is usually called the «Pseudo-Code», that is the rewriting of the initial legal text in an intermediated controlled natural language (English) that makes explicit logical structures and connectors, while keeping as much as possible of the original semantics. This is the stage that has to be supervised by legal experts. During this stage, specific guidelines were followed, taking into account the need for further processing of the representation. More in detail:

a) the knowledge engineer tried as much as possible to retain all the original sentences in such a way that a rule (or a set of rules) matches a sentence (or a set of sentences);

b) the connectives (and, or, not) were modeled by repeating the relevant part of the sentence or creating a new sentence;

c) the relatives (which, who, etc.) were developed and explicated;

d) pronouns were replaced by the nouns or lexical expressions they referred to;

e) irrelevant terms and repetitions were suppressed;

f) where necessary, sentences were modified to get a simpler syntax;

g) whenever possible, use was made of equivalent terms to minimize the number of derived predicate names;

h) sequences of words were replaced by single terms whenever compound words were found.

In the second stage of the work, we adopted a one-step process which, starting from the "pseudo-code", reached up to formalization in Prolog rules.

This stage was broken into three successive steps:

Step 1 – On the basis of an initial analysis of the list of nouns, verbs and adjectives contained in the source law texts, relevant lexical terms were identified, and a basic taxonomy was developed of the concepts of PIL, their properties and relations.

Step 2 – It was evaluated whether to include these terms as Prolog predicate names or arguments. Since a lexical term can be formalized as a different object type in the logical rules - predicates with different arities, functions, constants, etc. - the taxonomy was developed, and where necessary updated, to maintain consistency during the translation of the whole corpus of legal texts.

Step 3 - The final formalization was carried out from pseudo-code to Prolog rules. We adopted the methodology described in D3.1, section 3.4. We also decided to introduce structural elements into the rule-base, in the form of predicates, to refer to corresponding structural elements in the source legislative text (chapters and sections). Consider for example the following fragment of rules, extracted from the representation of the Brussels Regulation, section 7:

```
hasJurisdiction7_1to7(Country, Court, ClaimId, brusselsRegulation):-
    hasJurisdiction7_1(Country, Court, ClaimId, brusselsRegulation);
    hasJurisdiction7_2(Country, Court, ClaimId, brusselsRegulation);
    hasJurisdiction7_3(Country, Court, ClaimId, brusselsRegulation);
    hasJurisdiction7_4(Country, Court, ClaimId, brusselsRegulation);
    hasJurisdiction7_5(Country, Court, ClaimId, brusselsRegulation);
    hasJurisdiction7_6(Country, Court, ClaimId, brusselsRegulation);
    hasJurisdiction7_7(Country, Court, ClaimId, brusselsRegulation).

hasJurisdiction7_1(Country, Court, ClaimId, brusselsRegulation):-
    claimObject(ClaimId, contract, ContractId),
    placeOfPerformance(Country, Court).

hasJurisdiction7_2(Country, Court, ClaimId, brusselsRegulation):-
    claimObject(ClaimId, tort),
    eventOccurredOrMay(Country, Court).
```

In this example, the introduction of predicates in the form `hasJurisdiction7_1, 7_2, /_3, etc`), has a twofold purpose: 1) to maintain the isomorphism of the representation, keeping a close connection between the structure of the source texts and the structure of the representation, and 2) to keep trace of the articles in the execution process, so as to simplify debugging and validation of the rule-base, and even more importantly, to improve the quality of explanations provided by the meta-interpreter (see section 2.2.3 below).

The same approach was adopted for the Rome I and Rome II Regulations, with the inclusion of an additional argument, Article, that is assigned to the article that verifies the Goal. For consistency, the article reference in the predicate name was kept in the Rome Regulations, and the Brussels representation will be updated.

In the code, comments are used mostly as references to internal development processes. Prolog can also build a documentation in HTML from a subset of the documentation. Some testing has been done to keep the original text embedded in the source code, so that the relevant code and legal text can be viewed swiftly.

The main testing for the codebase is based on a small number of cases, and is automated via a Python wrapper.

2.2.2. Defeasibility and exceptions

It is commonly agreed that legal reasoning is defeasible reasoning. In the EU PIL corpus, defeasibility occurs by means of expressions such as: "unless proved otherwise"; "unless otherwise agreed"; "unless …"; "except in those cases"; "with the exception of"; "subject to…"; "notwithstanding"; etc.

In Prolog, defeasibility is usually managed by using negation by failure (See Deliverable 3.1, section 3.4.3). Hence, each of the above expressions has to be accounted for by using negation by failure.

The main non-ISO structure in the rule-base code is the meta-interpreter. A Prolog meta-interpreter is a Prolog program that takes a Prolog goal and another Prolog program, then proceeds to attempt to prove the goal against the second Prolog program, according to the rules given by the first program.

At a simple level, the meta-interpreter could be simply `prove(Goal) :- call(Goal)`, but this instruction would not add much information, since it would simply tell the main Prolog interpreter that to prove a goal it must call and verify that goal.

Below we present a revised version of the prototype meta-interpreter presented in Deliverable 3.1, section 3.4.4. We built this simple (but expandable) meta-interpreter to generate a proof tree tracing the execution of the program by printing a log of the evaluated predicates. This is our a first attempt at providing a user-friendly explanation functionality. Moreover, it allows further integration with other systems/languages by exposing the output with a stable API that can be integrated in / or communicate with the Decision Support Module of the InterLex Platform.

A goal that is found to be true during the exploration of the proof tree (i.e. an asserted predicate) is a fact, and is presented as such by the meta-interpreter.

```
% Metainterpreter
solve(true, [fact]) :- !.
solve((A,B), Result) :- !, solve(A, ARes), solve(B, BRes),
                        append(ARes, BRes, Result).
solve((A;B), Result) :- solve(A, Result); solve(B, Result).
```

```
solve(member(A,B), [member(A, B)]) :- !, call(member(A,B)).
solve(\+(A), [not(A)]) :- !, call(\+(A)).
solve((A)\=(B), [doNotUnify(A, B)]) :- !, call((A)\=(B)).
solve(A, [A|[Res]]) :- clause(A,B), solve(B, Res).
```

2.3. USE OF THE RULE-BASE

In this section, we present two examples showing the use of the SWI-Prolog engine in combination with the rule-base and a set of facts representing hypothetical scenarios involving the application of private international law rules for EU law, and national (Italian) legislation and case law.

2.3.1. Use case scenario for EU private international law: provision of services

Let us consider the following hypothetical scenario:

Silva Trade, a (natural) person located in Luxembourg, signs a contract with WoodFloor, a company located in Turin, Italy: Woodfloor agrees to provide its services to Silva in the cities of Vienna and Paris, and Silva agrees to pay a fee for the services. However, services are not provided as expected by Silva, and she refuses to pay the fee. Woodfloor decides to sue Silva.

The question to be assessed is: which court has jurisdiction over the claim?

In Prolog, the above facts may be represented as follows:

```
assert(claimMatter(claimId7_1, civilCommercial)).
assert(claimObject(claimId7_1, contract, contractId7_1)).
assert(contractTypeConsideration(contractId7_1, consumer)).
assert(contractType(contractId7_1, provisionOfServices)).
assert(personDomicile(silvaTrade, luxembourg, _)).
assert(personDomicile(woodFloor, italy, turin)).
assert(personNature(silvaTrade, natural)).
assert(personNature(woodFloor, legal)).
assert(personRole(silvaTrade, claimId7_1, defendant)).
assert(personRole(woodFloor, claimId7_1, claimant)).
assert(personType(silvaTrade, consumer)).
assert(placeOfProvision(austria, vienna)).
assert(placeOfProvision(france, paris)).
```

The query will be represented as follows:

```
hasJurisdiction(Country, Court, claimId7_1, brusselsRegulation).
```

The Prolog system provides three alternative solutions for the query:

```
luxembourg, _ ; france, paris ; austria, vienna
```

In order to verify the top-level goal of this chain (`hasJurisdiction(luxembourg, _1596, claimId7_1, brusselsRegulation`), Prolog has to verify its main conditions. The meta-interpreter collects such conditions at one level below the top goal, as part of a nested list:

```
[hasJurisdiction(luxembourg, Court, claimId7_1, brusselsRegulation),
[brusselsRegulationApplies(claimId7_1, brusselsRegulation),
[not(exception(brusselsRegulationApplies(claimId7_1, brusselsRegulation),
9548)), claimMatter(claimId7_1, civilCommercial), [fact]],
not(exception(hasGeneralJurisdiction, 9484, claimId7_1)),
hasGeneralJurisdiction(luxembourg, Court, claimId7_1, brusselsRegulation),
[personRole(silvaTrade, claimId7_1, defendant), [fact],
personDomicile(silvaTrade, luxembourg, Court), [fact],
memberState(luxembourg), [fact]]]]
```

Such an initial trace provided by the meta-interpreter is then formatted by the Python wrapper as follows:

```
TOP GOAL: luxembourg, 1
-> hasJurisdiction(luxembourg, _1596, claimId7_1, brusselsRegulation)
1)  -> brusselsRegulationApplies(claimId7_1, brusselsRegulation)
        -> not(exception(brusselsRegulationApplies(claimId7_1,
                                    brusselsRegulation), _1710))
        -> claimMatter(claimId7_1, civilCommercial)
            -> fact
2)  -> not(exception(hasGeneralJurisdiction, _1646, claimId7_1))
3)  -> hasGeneralJurisdiction(luxembourg, _1596, claimId7_1,
                              brusselsRegulation)
        -> personRole(silvaTrade, claimId7_1, defendant)
            -> fact
        -> personDomicile(silvaTrade, luxembourg, _1596)
            -> fact
        -> memberState(luxembourg)
            -> fact
```

In this explanation, the Prolog engine correctly verifies the main goal, with the value country= luxembourg, and court= unknown (_). The system then displays the predicates it verified to reach that goal (identified as `1)`, `2)`, and `3)` in the code lines above), indented to represent their logical chaining. The same happens to every intermediate goal that may need to be verified (in this case, the sub-goals are those identified as `1)`, `2)`, and `3)`, and their immediate sub-sub-goals).

The bottom-level statements that are verified by the Prolog engine correspond to the facts asserted at the beginning of the example.

2.3.2. Use case scenario for Italian private international law: prorogation of jurisdiction

Let us consider the following hypothetical scenario:

Bob, an entrepreneur based in Italy, starts proceedings in Italy against Ms Alice, Bob's former agent, asking for the restitution of an indemnity for termination of the agency contract paid by Bob to Alice. Alice disputes the jurisdiction of the Italian judge because the agency contract between Bob and Alice contained a choice of jurisdiction clause favouring the German court of Berlin. Bob, however, states that according to article 4.2 of the Italian PIL law, Italian jurisdiction cannot be derogated, since the controversy concerns a non-waivable right. Alice states instead that article 25 of the Brussels Regulation shall prevail over Italian law, so that Italian jurisdiction can be derogated.[3]

The question to be assessed is: which court has jurisdiction over the claim?

In Prolog, the above facts may be represented as follows:

```
assert(claimMatter(claim4_2, civilCommercial)).

assert(agreedJurisdiction(france, paris, claim4_2)).

assert(agreementCreationType(claim4_2, written)).

assert(personRole(alice, claim4_2, defendant)).

assert(personRole(bob, claim4_2, claimant)).

assert(personDomicile(bob, italy, milan)).

assert(agreementCreationType(claim4_2, writing)).

assert(agreedJurisdiction(germany, berlin, Claim4_2)).
```

If both the parties have their domicile in Italy, Italian PIL law will apply and an Italian court shall have jurisdiction. To test the solution in Prolog, we add the fact that Alice (the defendant) is domiciled in Italy:

```
assert(personDomicile(alice, italy, rome)).
```

Then we can call the query concerning the jurisdiction:

```
hasJurisdiction(Article, Country, Court, claim4_2, nationalLaw(italy)).
```

And the result will be:

```
Article = article4_1,

Country = italy,

Court = _ ;
```

[3] The example is modelled on the basis of the representation of the Italian PIL law and analysis of the Italian case law of the Court of Cassation (Cass., Sez. U., 10/05/2019, n.12585; and Cass., Sez. U., 14/2/2011, n. 3568).

The jurisdiction is set to an Italian court because Italian law would only accept an exception to this rule if the claim regarded waivable rights (which are not articulated in the facts asserted).

However, if the parties live in different countries, then the Brussels Regulation will determine the jurisdiction according to the rules of article 25. To test the solution in Prolog, we add the fact that Alice (the defendant) is domiciled in a country other than Italy, for example in Paris, France:

```
assert(personDomicile(alice, france, paris)).
```

If now we call the EU law:

```
hasJurisdiction(Article, Country, Court, claim4_2, brusselsRegulation).
```

the result will be:

```
Article = article25,

Country = germany,

Court = berlin ;
```

because EU law does not preclude derogating jurisdiction due to non-waivable rights.

3. Conclusions

The complexity of source norms has sometimes limited the isomorphism of the representation, particularly in the management of norms constituting implicit exceptions to general norms. However, the issue has been solved with the solution proposed in Deliverable D3.1 and applied as in section 2.2.2 of the present Deliverable.

Besides, we were aware that the domain of PIL would be particularly difficult to formalize, especially when compared to those domains that are traditional fields of application of knowledge-based systems in law, such as tax legislation, administrative regulations, provision of benefits, etc.

It is true that tax legislation is commonly considered “complex” because of the high number of exceptions and requirements to be verified to reach a conclusion. However, tax rules are quite easy to represent because they do not need any particular work of interpretation by a legal expert.

On the other hand, the PIL domain contains a very high number of concepts that lack an authoritative definition set by the lawmakers, and which may therefore be considered as “open texture” concepts (as defined in legal philosophy by Hart [4] and discussed in relation to legal information systems by [5]). Their meaning should be identified by searching in multiple sources case law and doctrinal interpretations.

Despite the high difficulty and complexity of representing private international law , the results presented in this report show how potentially any field of the law may be represented in the form of rules using the adopted methodology – with opportune adaptations – and transposed into a platform such as InterLex.

The extension of the representation to integrate Italian and Czech national legislation and case law has been carried out using the modular approach described in section 2.1. We are confident that such a modular approach will make it possible to further extend the representation to cover additional national legislation and, if needed, accommodate amendments and other changes to national legal instruments, without impacting on the core structure of the representation.

References

[1] U. Magnus, Brussels I Regulation (European Commentaries on Private International Law), Sellier european law publishers, 2007.

[2] G. Callies, Rome Regulations: Commentary on the European Rules of the Conflict of Laws, Wolters Kluwer Law & Business, 2015.

[3] F. Mosconi and C. Campiglio, Diritto internazionale privato e processuale, Pavia: UTET, 2015.

[4] H. Hart, The Concept of Law, Oxford: Clarendon Press, 1961.

[5] T. a. V. P. Bench-Capon, "Open texture and ontologies in legal information systems," in *Database and Expert Systems Applications, 1997. Proceedings., Eighth International Workshop on*, 2002.

Appendix

Complete Prolog rulebase: